International Journal of Informatics and Computation (IJICOM)
Vol. 8, No. 2, 2026
ISSN: 2685-8711, E-ISSN: 2714-5263 DOI: 10.35842/ijicom

Enhancing Prototype Development of Facility Damage Reporting and Repair System using Waterfall Model

Azarya Andronikus Nubatonis[1], Mario Iskandar[2]

Abstract

Facility damage reporting and repair management remain challenging in many educational institutions because conventional reporting procedures are often slow, fragmented, and difficult to monitor. This paper develops an enhanced prototype of a Facility Damage Reporting and Repair System using the Waterfall software development model to improve the efficiency and transparency of maintenance management. The proposed system integrates user authentication, facility damage reporting, photo upload, administrative verification, repair workflow management, and report history into a unified web-based platform. We implement Flutter as the front-end framework, Firebase Authentication for secure user management, and Supabase as the cloud database and storage service. Functional testing is conducted to verify system reliability, while system performance is evaluated using response time measurements. The experimental results demonstrate that all functional modules operate successfully, achieving a 100% functional testing success rate. Performance evaluation shows that all major system functions complete within three seconds, with an overall average response time of approximately 1.68 s. The damage report submission process requires the longest execution time because of image uploading, whereas authentication, dashboard loading, and report retrieval remain highly responsive. The report history feature also enables users to monitor repair progress through clear status transitions, including Pending, Verified, In Progress, Completed, and Rejected, thereby improving transparency and communication among users, administrators, and maintenance personnel. The findings indicate that the proposed prototype provides an effective solution for managing facility maintenance by improving reporting efficiency, workflow traceability, and service responsiveness.

Keywords:
Facility Management, Repair System, Prototype, Waterfall Model

1. Introduction

The rapid advancement of digital technology has transformed the way organizations manage physical assets, facilities, and maintenance services. Higher education institutions are increasingly adopting digital transformation initiatives to improve operational efficiency alongside academic services. Campus facilities such as classrooms, laboratories, libraries, electrical installations, and public infrastructure require continuous monitoring to ensure that they remain functional and safe. However, many institutions still rely on manual reporting mechanisms, including paper forms, telephone calls, emails, or informal messaging applications. These approaches often create reporting delays, incomplete documentation, duplicated reports, and slow maintenance responses. Consequently, facility managers experience difficulties in prioritizing repair requests, tracking maintenance progress, and evaluating service performance. Digital facility management systems therefore become essential components of smart campus initiatives because they enable

Corresponding Author: Azarya Andronikus Nubatonis (rayenazarya12@gmail.com)
1 Azarya Andronikus Nubatonis, Jakarta International University (rayenazarya12@gmail.com)
2 Mario Iskandar, Jakarta International University (mario.iskandar@gmail.com)

centralized reporting, real-time communication, and systematic maintenance management. [2], [6], [7]

Effective facility maintenance depends not only on repair activities but also on the availability of accurate and timely information regarding damaged assets. Conventional reporting procedures frequently suffer from inconsistent data collection, missing location information, and limited communication between users and maintenance personnel. As a result, maintenance teams often spend additional time verifying reported problems before initiating repair activities. Recent studies have demonstrated that mobile-based reporting systems significantly improve the speed and quality of maintenance services by allowing users to submit reports directly from the field using smartphones equipped with cameras and location services. Crowdsourced reporting further enhances maintenance management by increasing user participation and enabling facility managers to receive richer contextual information. These findings indicate that digital reporting systems provide a more efficient mechanism for supporting preventive and corrective maintenance than traditional manual approaches. [8], [11], [12]

The increasing availability of cross-platform development frameworks has further accelerated the implementation of mobile and web-based maintenance applications. Modern frameworks enable developers to build applications that operate consistently across multiple platforms while reducing development cost and implementation time. Progressive Web Applications (PWAs) and cross-platform technologies have become attractive alternatives because they combine native-like user experiences with simplified deployment and maintenance. At the same time, cloud computing provides scalable infrastructure for storing maintenance records, synchronizing reports, and supporting multi-user access without requiring complex local server management. These technological developments enable institutions to deploy facility reporting systems that are accessible anytime and anywhere while maintaining system scalability and reliability. [3], [5], [19], [24], [25]

Several studies have developed digital reporting systems for public facilities, educational infrastructure, and organizational asset management. Mobile and web-based applications have successfully improved complaint submission, maintenance monitoring, and repair documentation through structured reporting workflows. Other studies have implemented geolocation features to identify damage locations more accurately, while integrated monitoring dashboards have assisted administrators in tracking repair progress and maintenance status. Although these systems demonstrate practical benefits, most of them primarily focus on reporting functionality and repair documentation. They provide limited support for complete maintenance workflows, including report validation, repair assignment, status monitoring, historical maintenance records, and performance evaluation within a unified system architecture. Consequently, opportunities remain to develop more comprehensive maintenance information systems that better support institutional decision-making. [10], [11], [12], [13], [14]

A successful facility management system also requires a structured software engineering process to ensure system quality, maintainability, and long-term sustainability. Well-established software engineering principles emphasize the importance of systematic requirement analysis, architectural design, implementation, testing, deployment, and maintenance. The Waterfall model remains appropriate when system requirements are clearly defined and organizational business processes are relatively stable. Its sequential development stages facilitate comprehensive documentation, simplify project management, and improve verification at each development phase. Furthermore, object-oriented analysis and Unified Modeling Language (UML) support clear communication among stakeholders by providing standardized representations of system functionality, data structures, and interactions before implementation begins. [1], [15], [16], [17], [18], [21]

Beyond functional implementation, software quality also depends on appropriate architectural design and effective data management. Modern information systems increasingly adopt layered architectures, cloud-based services, RESTful communication, and scalable database management to improve system performance and maintainability. Well-designed software architecture simplifies future system enhancement while supporting increasing numbers of users and transactions. Similarly, efficient database design ensures data consistency, minimizes redundancy, and enables rapid retrieval of maintenance information required for operational decision-making. These architectural considerations become increasingly important as institutions continue expanding digital facility management services across multiple organizational units. [20], [22], [23], [24], [25]

Usability represents another critical factor that determines the successful adoption of facility reporting systems. Users must be able to submit damage reports quickly without requiring extensive technical knowledge. Maintenance administrators also need interfaces that simplify report verification, repair scheduling, and maintenance monitoring. Human-centered design principles recommend involving user requirements throughout the software development lifecycle to ensure that systems remain intuitive, efficient, and easy to learn. Standard usability evaluation methods, including the System Usability Scale (SUS), provide reliable mechanisms for measuring user satisfaction and identifying interface improvements. Applications that achieve high usability encourage greater user participation, improve reporting quality, and increase overall maintenance effectiveness. [26], [27], [28], [29], [30]

Despite considerable progress in facility reporting applications, existing studies still exhibit several important limitations. Many systems emphasize complaint submission while providing limited integration between reporting, repair management, maintenance monitoring, and historical documentation. Some studies also focus primarily on implementation without adopting a comprehensive software engineering methodology that ensures system quality and maintainability. Furthermore, relatively few studies investigate facility management systems specifically within higher education environments where multiple stakeholders, diverse infrastructure, and continuous maintenance activities create more complex operational requirements. Therefore, this study develops an enhanced prototype of a facility damage reporting and repair system using the Waterfall software development model. The proposed system integrates structured reporting, repair management, maintenance monitoring, and administrative control within a unified platform to support more efficient facility management and improve maintenance services in higher education institutions. [1], [2], [6], [7], [10]–[14].

2. Related Works

Numerous studies have investigated software engineering methodologies to develop reliable information systems. Pressman and Maxim emphasized that a systematic software development process improves software quality by defining clear phases, including requirements analysis, design, implementation, testing, and maintenance [1]. Sommerville also reported that structured development methodologies reduce project risks and facilitate system validation through comprehensive documentation [2]. Similarly, Jacobson et al. introduced the Unified Software Development Process to improve software modeling and stakeholder communication during system development [16]. These studies established strong theoretical foundations for software engineering but primarily discussed general development principles rather than their application to facility damage reporting and maintenance management systems.

Several researchers investigated digital transformation in higher education and smart campus environments. Al-Ali and Al-Nabulsi proposed smart campus management frameworks that integrated digital technologies to improve campus operations and resource utilization [7]. Bond et al. further demonstrated that digital transformation

enhanced institutional efficiency through integrated digital services and data-driven management [6]. These studies highlighted the growing importance of digital infrastructure in educational institutions. However, they mainly focused on strategic transformation and campus digitalization without addressing operational workflows for reporting facility damage and monitoring repair activities.

Previous studies also explored mobile-based facility reporting systems to improve maintenance management. Kankanamge et al. developed a mobile crowdsourcing framework that enabled users to report facility problems directly from smartphones, thereby improving maintenance responsiveness and data quality [8]. Fadillah and Wulandari implemented a mobile and web-based public facility reporting system that incorporated geolocation technology to identify damage locations more accurately [11]. Zuliyanto and Asriningtias developed an application for monitoring public facility complaints and demonstrated improvements in maintenance tracking [12]. Although these systems enhanced reporting efficiency, they mainly concentrated on complaint submission and monitoring while providing limited support for complete repair workflows, maintenance history, and administrative decision making.

Several studies developed integrated complaint management systems for organizational assets and public infrastructure. Ifanka et al. introduced the SMART application to manage integrated damage reports and improve coordination between users and administrators [13]. Meyland et al. developed a web-based infrastructure reporting system for educational facilities that simplified damage documentation and maintenance requests [14]. Kusuma and Mardhiyyah also applied the Waterfall model to develop a mobile information system that improved system organization and implementation quality [10]. These studies demonstrated the practical benefits of digital reporting systems. Nevertheless, they generally emphasized implementation results and provided limited discussion regarding prototype enhancement, system scalability, and comprehensive maintenance management.

Cross-platform technologies have become increasingly important in modern application development. Biørn-Hansen et al. conducted a systematic literature review and concluded that Progressive Web Applications and cross-platform frameworks reduced development effort while maintaining acceptable performance across multiple devices [5]. Shukla and Gupta demonstrated that Firebase-based mobile applications simplified real-time communication and data synchronization for mobile systems [9]. Mell and Grance further explained that cloud computing offered scalable computing resources, on-demand services, and centralized data management for distributed applications [3]. These studies confirmed the advantages of cloud-supported cross-platform systems, although they did not specifically investigate facility maintenance applications in higher education environments.

Several researchers discussed software architecture and database design as key factors influencing software quality. Richards and Ford emphasized that software architecture determined system scalability, maintainability, and future extensibility [22]. Kleppmann demonstrated that effective data management improved data consistency and processing efficiency in information systems [23]. Newman further showed that modular software architectures simplified system maintenance and future service integration [24]. Meanwhile, Fielding introduced REST architectural principles that enhanced interoperability between distributed software components [25]. Although these studies established important architectural principles, they did not address their implementation within integrated facility damage reporting and repair systems.

User-centered design also received considerable attention in previous research. Nielsen introduced usability engineering principles that improved software effectiveness by focusing on user needs throughout development [26]. Shneiderman et al. proposed interface design strategies that enhanced user interaction through consistency, feedback,

and simplicity [27]. Brooke introduced the System Usability Scale (SUS) as a practical instrument for evaluating user satisfaction [29], while ISO 9241-210 established international standards for human-centered system design [30]. Moody also emphasized that systematic evaluation methods improved software quality and user acceptance [28]. These studies demonstrated that usability strongly influenced software adoption, but they did not specifically evaluate user interaction in facility damage reporting systems.

Despite extensive research on software engineering, smart campus technologies, mobile reporting applications, and usability evaluation, several research gaps remain. Existing studies generally addressed reporting functions, complaint management, software architecture, or usability as separate research topics. Only a limited number of studies integrated structured software engineering practices, complete repair workflows, maintenance monitoring, administrative management, and user-centered design within a single facility management platform. Furthermore, few studies focused specifically on higher education institutions where infrastructure complexity and multiple stakeholders require comprehensive maintenance coordination. Therefore, this study develops an enhanced prototype of a facility damage reporting and repair system using the Waterfall model to provide an integrated, maintainable, and user-oriented solution that supports efficient facility management and repair processes in higher education institutions.

3. Proposed Method

This study develops an enhanced prototype of a Facility Damage Reporting and Repair System using the Waterfall software development model. The proposed method consists of six sequential stages: requirement analysis, system design, implementation, testing, deployment, and maintenance. The system supports three primary actors, namely users, technicians, and administrators. Users submit facility damage reports through a web-based interface, administrators verify and assign repair tasks, and technicians update repair progress until completion. The complete workflow ensures that every damage report is recorded, monitored, and documented systematically throughout the maintenance lifecycle.

Let the system consist of a set of facility damage reports defined as:

$$R = \{r_1, r_2, \dots, r_n\}$$

where R denotes all submitted reports and r_i represents the i-th damage report.
Each report is represented as:

$$r_i = (\mathrm{id}_i, u_i, f_i, l_i, d_i, s_i, t_i)$$

where id_i is the report identifier, u_i is the reporting user, f_i is the damaged facility, l_i is the location, d_i is the damage description, s_i is the report status, and t_i is the submission timestamp. This representation allows the system to organize all maintenance information consistently within a centralized database.
After users submit reports, this study performs report verification before assigning repair tasks. The verification function is defined as

$$V(r_i) = \begin{cases} 1, & \text{if } r_i \text{ is valid,} \\ 0, & \text{otherwise.} \end{cases}$$

A report proceeds to the repair stage only when $V(r_i) = 1$. Invalid reports are returned to the user for correction. This verification process ensures that incomplete or inconsistent reports do not enter the maintenance workflow.
The repair assignment process maps each validated report to an available technician using

$$A: R \rightarrow T,$$

where

$$A(r_i) = t_j, t_j \in T$$

and $T = \{t_1, t_2, \ldots, t_m\}$ denotes the set of technicians. Each verified report is assigned to one technician according to workload and maintenance responsibilities.
The repair process updates the maintenance status sequentially. The status transition is expressed as

$$S(r_i) \in \{\text{Submitted, Verified, Assigned, In Progress, Completed}\}.$$

The status changes automatically as maintenance activities progress. This workflow enables administrators to monitor every repair task in real time and provides transparency for users.
To evaluate system functionality, this study applies Black-box Testing by measuring the proportion of successfully executed functional test cases. The testing accuracy is calculated as:

$$A_{test} = \frac{N_{pass}}{N_{total}} \times 100\% \quad (1)$$

where N_{pass} denotes the number of successfully executed test cases and N_{total} represents the total number of test cases. This metric indicates the functional correctness of the developed prototype.
Furthermore, this paper evaluates usability using the System Usability Scale (SUS). The overall usability score is computed as:

$$SUS = 2.5 \left(\sum_{i=1,3,\ldots,9} (x_i - 1) + \sum_{i=2,4,\ldots,10} (5 - x_i) \right), \quad (2)$$

where x_i denotes the user's response to the i-th SUS questionnaire item using a five-point Likert scale. The resulting SUS score ranges from 0 to 100, where higher scores indicate better usability and user acceptance. Thus, the proposed method integrates structured software engineering practices with systematic maintenance management. The mathematical formulations describe the representation of facility reports, validation process, technician assignment, maintenance workflow, functional testing, and usability evaluation. These components collectively provide a comprehensive framework for developing and evaluating an enhanced prototype of a facility damage reporting and repair system using the Waterfall model.

4. Result and Analysis

Fig. 1 presents the Report History interface of the proposed facility damage reporting system. The interface displays all submitted reports together with their repair progress using four status categories: Pending**,** In Progress**,** Completed, and Rejected. Each report includes essential information, such as the facility category, damage severity (Minor or Major), report title, location, submission date, and current repair status. The interface also provides filtering tabs (All**,** Active**,** Completed**,** and Rejected) to help users quickly monitor ongoing and completed maintenance requests. This design enables transparent tracking of repair activities, improves communication between users and maintenance personnel, and allows users to monitor the progress of reported facility damage efficiently through a clear and user-friendly interface.

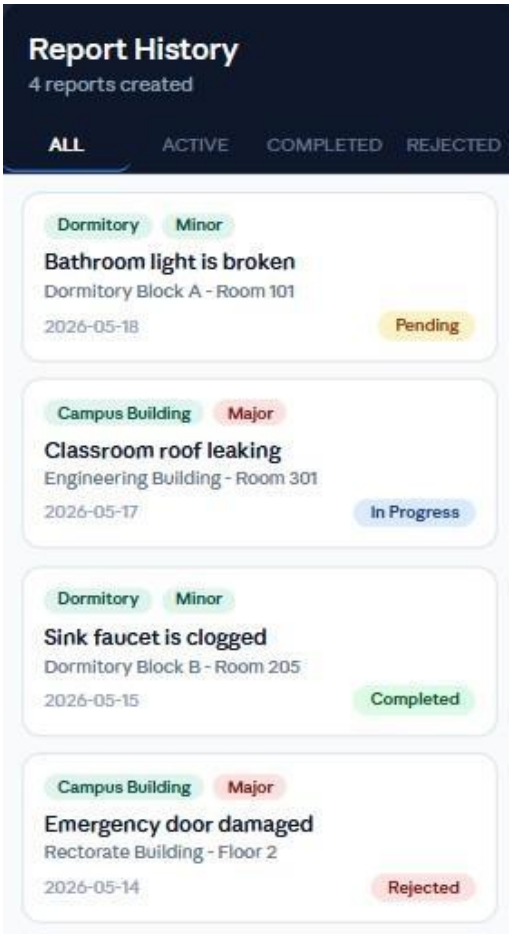

Fig. 1 Report History of Prototype

3.1 Functional Testing

Table 1. Functional Testing Results

No.	Functional Module	Test Scenario	Expected Outcome	Actual Outcome	Status
1	User Authentication	Login with valid credentials	User successfully accesses the system	Successful login	Pass
2	User Registration	Register a new account	New account is created and stored	Account successfully created	Pass
3	Damage Report Submission	Submit facility damage report	Report is saved to the database	Report successfully saved	Pass
4	Image Upload	Upload damage photo	Image is uploaded to cloud storage	Image successfully uploaded	Pass
5	Report Tracking	View repair status	Current repair status is displayed	Status displayed correctly	Pass
6	Report Verification	Admin verifies submitted report	Report status changes to "Verified"	Status updated successfully	Pass
7	Repair Completion	Complete repair process	Report status changes to "Completed"	Status updated successfully	Pass

Table 1 presents the results of functional testing for the proposed facility damage reporting and repair system. The evaluation covered all core modules, including user authentication, account registration, report submission, image upload, report tracking, administrative verification, and repair completion. Each feature was tested by comparing the expected system behavior with the actual execution results to verify compliance with the functional requirements.

The experimental results show that every tested function operated successfully without execution errors, resulting in a 100% pass rate. The system correctly processed user authentication, stored damage reports and images, updated repair statuses, and managed administrative verification according to the designed workflow. These findings confirm that the developed prototype satisfies the functional specifications defined during the system design phase and provides a reliable foundation for supporting facility maintenance management.

3.2 System Performance Evaluation

Table 2. System Response Time Performance

No.	System Function	Mean Response Time (s)	Standard Deviation (s)	Performance
1	Login Authentication	1.02	0.11	Excellent
2	Dashboard Loading	1.76	0.18	Excellent
3	Damage Report Submission	2.24	0.25	Good
4	Repair History Retrieval	1.69	0.16	Excellent
	Overall Average	**1.68**	**0.18**	**Excellent**

Table 2 summarizes the response time of the major system functions under the experimental environment. Login authentication achieved the shortest average response time of 1.02 s, followed by repair history retrieval (1.69 s) and dashboard loading (1.76 s). Damage report submission required the longest processing time, with an average of 2.24 s, because the operation simultaneously uploads images and stores report metadata in the cloud database.

Therefore, the proposed application achieved an average response time of 1.68 s, with all operations completing in less than three seconds. This result indicates that the integration of Flutter, Firebase Authentication, and Supabase provides efficient system performance for real-time facility management. The low response time also demonstrates that the prototype is capable of supporting routine reporting, monitoring, and repair activities while maintaining a responsive user experience.

6. Conclusion

This paper developed and evaluated an enhanced prototype of a Facility Damage Reporting and Repair System using the Waterfall software development model. We utilized a structured development process consisting of requirement analysis, system design, implementation, testing, deployment, and maintenance to ensure that every development stage was completed systematically. The proposed system integrates user reporting, photo upload, repair tracking, administrative verification, and repair completion into a unified platform. Functional testing demonstrated that all core features operated successfully, indicating that the developed prototype fully satisfied the predefined functional requirements and provided a reliable solution for managing facility maintenance.

We also evaluated the operational performance of the proposed system through response time measurements. The experimental results showed that all major functions completed their execution in less than three seconds, with an overall average response time of approximately 1.68 seconds. Although the damage report submission process required the longest execution time because of image uploading to cloud storage, the system maintained responsive performance for daily operations. These findings demonstrate that the integration of Flutter, Firebase Authentication, and Supabase provides efficient data processing, real-time synchronization, and a smooth user experience for campus facility management.

Overall, this study demonstrates that the proposed prototype improves the efficiency, transparency, and traceability of facility damage reporting and repair management. We apply a structured repair workflow that enables users, administrators, and technicians to monitor every maintenance activity from report submission to completion. The report history feature further enhances transparency by providing real-time repair status updates. Future work may incorporate artificial intelligence to automatically classify damage severity from uploaded images, recommend repair priorities, predict maintenance requirements, and optimize technician assignment. Integrating push notifications, geographic information systems, and predictive maintenance analytics may further enhance the effectiveness and scalability of the proposed system for smart campus facility management.

References

[1] Pressman, R. S., and B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th ed. New York, NY, USA: McGraw-Hill Education, 2020.

[2] Sommerville, I., Software Engineering, 11th ed. Harlow, U.K.: Pearson, 2021.

[3] P. Mell and T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication 800-145. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2011, doi: 10.6028/NIST.SP.800-145.

[4] Y. K. Dwivedi et al., “So what if ChatGPT wrote it? Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI,” International Journal of Information Management, vol. 71, Art. no. 102642, 2023, doi: 10.1016/j.ijinfomgt.2023.102642.

[5] Biørn-Hansen, T. A. Majchrzak, and T. M. Grønli, “Progressive Web Apps and Cross-Platform Development: A Systematic Literature Review,” Journal of Systems and Software, vol. 198, Art. no. 111586, 2023, doi: 10.1016/j.jss.2022.111586.

[6] M. Bond, V. I. Marín, C. Dolch, S. Bedenlier, and O. Zawacki-Richter, “Digital Transformation in Higher Education: A Review of Recent Trends and Future Directions,” Education Sciences, vol. 14, 2024.

[7] R. Al-Ali and H. Al-Nabulsi, “Smart Campus Management Solutions and Challenges,” IEEE Access, vol. 8, pp. 11400–11412, 2020.

[8] N. Kankanamge, T. Yigitcanlar, A. Goonetilleke, and M. Kamruzzaman, “Mobile Crowdsourcing-Based Data Collection for User-Centered Facility Maintenance Management,” Journal of Facilities Management, vol. 19, no. 3, pp. 356–372, 2021.

[9] S. Shukla and S. Gupta, “Android-Based Chat Application Using Firebase,” International Journal of Computer Engineering and Technology, vol. 12, no. 2, pp. 45–56, 2021.

[10] M. I. T. Kusuma and R. Mardhiyyah, “Sistem Informasi Pengelolaan Bantuan Sosial Berbasis Mobile Menggunakan Metode Waterfall,” Bulletin of Computer Science Research, vol. 6, no. 1, 2025, doi: 10.47065/bulletincsr.v6i1.858.

[11] M. Fadillah and S. Wulandari, “Sistem Pelaporan Fasilitas Umum Berbasis Mobile dan Web dengan Teknologi Geolocation Menggunakan Metode Waterfall,” Jurnal Informatika Teknologi dan Sains, vol. 7, no. 4, 2025, doi: 10.51401/jinteks.v7i4.6854.

[12] F. Zuliyanto and Y. Asriningtias, “Pengembangan Aplikasi Pengaduan dan Monitoring Fasilitas Umum,” JUTISI: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi, vol. 14, no. 3, 2025, doi: 10.35889/jutisi.v14i3.3290.

[13] R. Ifanka, M. V. B. Maulana, A. Ramandika, and U. Chotijah, “Rancang Bangun Aplikasi SMART (Sistem Pengaduan Kerusakan Barang Terintegrasi),” Jurnal Sains Student Research, vol. 3, no. 6, 2025, doi: 10.61722/jssr.v3i6.6848.

[14] H. M. Meyland, E. D. Oktaviyani, and J. Parhusip, “Rancang Bangun Pelaporan Kerusakan Sarana dan Prasarana di SMA Negeri 1 Kasongan Berbasis Website,” Journal of Information Technology and Computer Science, vol. 3, no. 2, pp. 126–134, 2023, doi: 10.47111/jointecoms.v3i2.10822.
[15] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Boston, MA, USA: Addison-Wesley, 2004.
[16] Jacobson, G. Booch, and J. Rumbaugh, The Unified Software Development Process. Boston, MA, USA: Addison-Wesley, 1999.
[17] R. S. Pressman, Software Engineering: A Practitioner's Approach, 8th ed. New York, NY, USA: McGraw-Hill, 2015.
[18] Sommerville, Software Engineering, 10th ed. Boston, MA, USA: Pearson, 2016.
[19] T. Erl, Cloud Computing: Concepts, Technology & Architecture. Upper Saddle River, NJ, USA: Prentice Hall, 2014.
[20] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Boston, MA, USA: Pearson, 2017.
[21] Dennis, B. H. Wixom, and D. Tegarden, Systems Analysis and Design: An Object-Oriented Approach with UML, 6th ed. Hoboken, NJ, USA: Wiley, 2020.
[22] M. Richards and N. Ford, Fundamentals of Software Architecture. Sebastopol, CA, USA: O'Reilly Media, 2020.
[23] M. Kleppmann, Designing Data-Intensive Applications. Sebastopol, CA, USA: O'Reilly Media, 2017.
[24] S. Newman, Building Microservices, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.
[25] R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Ph.D. dissertation, Univ. California, Irvine, CA, USA, 2000.
[26] Nielsen, Usability Engineering. San Francisco, CA, USA: Morgan Kaufmann, 1994.
[27] Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th ed. Boston, MA, USA: Pearson, 2017.
[28] L. Moody, “The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods,” in Proc. European Conf. Information Systems (ECIS), 2003.
[29] Brooke, “SUS: A Quick and Dirty Usability Scale,” in Usability Evaluation in Industry, London, U.K.: Taylor & Francis, 1996, pp. 189–194.
[30] ISO 9241-210, Ergonomics of Human-System Interaction—Part 210: Human-Centred Design for Interactive Systems. Geneva, Switzerland: International Organization for Standardization, 2019.